The Representation and Matching of Categorical Shape

Ali Shokoufandeh
Lars Bretzner
Diego Macrini
M. Fatih Demirci
Clas Jönsson
and
Sven Dickinson

Technical Report DU-CS-05-10
Department of Computer Science
Drexel University
Philadelphia, PA 19104
July, 2005

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **JUL 2005**	2. REPORT TYPE	3. DATES COVERED **00-00-2005 to 00-00-2005**

4. TITLE AND SUBTITLE **The Representation and Matching of Categorical Shape**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Drexel University,Department of Computer Science,Philadelphia,PA,19104**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
We present a framework for categorical shape recognition. The coarse shape of an object is captured by a multiscale blob decomposition, representing the compact and elongated parts of an object at appropriate scales. These parts, in turn, map to nodes in a directed acyclic graph, in which edges encode both semantic relations (parent/child or sibling) as well as geometric relations. Given two image descriptions each represented as a directed acyclic graph, we draw on spectral graph theory to derive a new algorithm for computing node correspondence in the presence of noise and occlusion. In computing correspondence, the similarity of two nodes is a function of their topological (graph) contexts, their geometric (relational) contexts and their node contents. We demonstrate the approach on the domain of view-based 3-D object recognition.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**42**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

The Representation and Matching of Categorical Shape

Ali Shokoufandeh[a] Lars Bretzner[b] Diego Macrini[c]

M. Fatih Demirci[a] Clas Jönsson[b] Sven Dickinson[c]

[a]*Department of Computer Science, Drexel University, Philadelphia, PA 19104, USA*

[b]*Computational Vision and Active Perception Laboratory, Department of Numerical Analysis and Computer Science, KTH, Stockholm, Sweden*

[c]*Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 3G4*

Abstract

We present a framework for categorical shape recognition. The coarse shape of an object is captured by a multiscale blob decomposition, representing the compact and elongated parts of an object at appropriate scales. These parts, in turn, map to nodes in a directed acyclic graph, in which edges encode both semantic relations (parent/child or sibling) as well as geometric relations. Given two image descriptions, each represented as a directed acyclic graph, we draw on spectral graph theory to derive a new algorithm for computing node correspondence in the presence of noise and occlusion. In computing correspondence, the similarity of two nodes is a function of their topological (graph) contexts, their geometric (relational) contexts, and their node contents. We demonstrate the approach on the domain of view-based 3-D object recognition.

Key words: generic object recognition, shape categorization, graph matching, scale spaces, spectral graph theory

1 Introduction

Object categorization has long been a goal of the object recognition community, with notable early work by Binford [2], Marr and Nishihara [28], Agin and Binford [1], Nevatia and Binford [31], and Brooks [4] attempting to construct and recognize prototypical object models based on their coarse shape structure. However, the representational gap between low-level image features and the abstract nature of the models was large, and the community lacked the computational infrastructure required to bridge this representational gap [18]. Instead, images were simplified and objects were textureless, so that extracted image features could map directly to salient model features. In the years to follow, such generic object recognition systems gave way to exemplar-based systems, first passing through highly constrained geometric (CAD) models, and on to today's generation of appearance-based models. Whether the images were moved closer to the models (earlier approaches) or the models moved closer to the images (later approaches), the representational gap has remained largely unaddressed.

The community is now returning to the problem of object categorization, using powerful machine learning techniques and new appearance-based features. However, appearance-based representations are not invariant to significant within-class appearance change, due to texture, surface markings, structural detail, or articulation. As a result, the categories are often very restricted, such as faces, cars, motorcycles, and specific species of animals. Accommodating significant within-class shape variation puts significant demands on a representation: 1) it must capture the coarse structure of an object; and 2) it must be local, in order to accommodate occlusion, clutter and noise. These criteria point to a structured shape description not unlike the ambitious part-based recognition frameworks proposed in the 70's and 80's. However, the representational gap facing these early systems still poses a major obstacle.

We begin by imposing a strong shape prior on the parts and relations making up an object. Specifically, we represent a 2-D object view as a multiscale blob decomposition, whose part vocabulary includes two types, compact blobs and elongated ridges, and whose relations also include two types, semantic (parent/child and sibling) and geometric. The detected qualitative parts and and their relations are captured in a directed acyclic graph, called a *blob graph*, in which nodes represent parts and edges capture relations.

Choosing a restricted vocabulary of parts helps us bridge the representational gap. Early generation systems started with low-level features such as edges, regions, and interest points, and were faced with the challenging task of grouping and abstracting them to form high-level parts, such as generalized cylinders. By severely restricting the part vocabulary, we construct a high-level part detector from simple, multiscale filter responses. Although the parts are simple and qualitative, their semantic and geometric relations add rich structure to the representation, yielding a shape representation that can be used to discriminate shape categories.

Any shortcut to extracting high-level part structure from low-level features, such as filter responses, will be prone to error. Thus, the recovered blob graph will be missing nodes/edges and will contain spurious nodes/edges, setting up a challenging inexact graph matching problem. In our previous work on matching rooted trees, [42], we drew on spectral graph theory to represent the coarse "shape" of a tree as a low-dimensional vector based on the eigenvalues of the tree's symmetric adjacency matrix. Our matching algorithm utilized these vectors in a coarse-to-fine matching strategy that found corresponding nodes. Although successfully applied to *shock trees*, the matching algorithm suffered from a number of limitations: 1) it could not handle the directed acyclic graph structure found in, for example, our multiscale blob graphs, e.g., a blob may have two parents; 2) it was restricted to matching hierarchical parent/child relations, and could not accommodate sibling relations, e.g., the geometric relations between blobs at a given scale; and 3) the matching algorithm was an approximation algorithm that could not ensure that hierarchical and sibling relations were preserved during matching, allowing, for example, two siblings (sharing a parent) in one tree to match two nodes in the other tree having an ancestor/descendant relationship.

We first extend our matching algorithm to handle directed acyclic graphs, drawing on our recent work in indexing hierarchical structures [40], in which we represent the coarse shape of a directed acyclic graph as a low-dimensional vector based on the eigenvalues of the DAG's antisymmetric adjacency matrix. Next, we introduce a notion of graph neighborhood context, allowing us to accommodate sibling relations, such as our blob graphs' geometric relations, into our matching algorithm. Like our vector encoding of hierarchical structure, local sibling structure is also encoded in a low-dimensional vector. Finally, we extend the matching algorithm to ensure that that both hierarchical and sibling relations are enforced during matching, yielding improved correspondence in the presence of noise and occlusion. The result is a far more powerful matching framework that's ideally suited to our multiscale blob graphs.

Following a review of related work in Section 2, we describe our qualitative shape representation in Section 3. Next, we describe our new matching algorithm in Section 4. In Section 5, we evaluate the approach on the domain of view-based 3-D object recognition. We discuss the limitations of the approach in Section 6, and draw conclusions in Section 7.

2 Related Work

There has been considerable effort devoted to both scale space theory and hierarchical structure matching, although much less effort has been devoted to combining the two paradigms. Coarse-to-fine image descriptions are plentiful, including work by Burt [5], Lindeberg [26], Simoncelli et al. [43], Mallat and Hwang [27], and Witkin and Tennenbaum [**?**]. Some have applied such models to directing visual attention, e.g., Tsotsos [46], Jägersand [17], Olshausen et al. [32], and Takaċs and Wechsler [44]. Although such descriptions are well suited for tasks such as compression, attention, or object localization, they often lose the detailed shape information required for object recognition.

Others have developed multi-scale image descriptions specifically for matching and recognition. Crowley and Sanderson [8,7,9] extracted peaks and ridges from a Laplacian pyramid and linked them together to form a tree structure. However, during the matching phase, little of the trees' topology or geometry was exploited to compute correspondence. Rao et al. [35] correlate a rigid, multiscale saliency map of the target object with the image. However, like any template-based approach, the technique is rather global, offering little invariance to occlusion or object deformation. In an effort to accommodate object deformation, Wiskott et al. apply elastic graph matching to a planar graph whose nodes are wavelet jets. Although their features are multi-scale, their representation is not hierarchical, and matching requires that the graphs be coarsely aligned in scale and image rotation [48]. A similar approach was applied to hand posture recognition by Triesch and von der Malsburg [45].

The representation of image features at multiple scales suggests a hierarchical graph representation, which can accommodate feature attributes in the nodes and relational attributes in the arcs. Although graph matching is a popular topic in the computer vision literature [12], including both inexact and exact graph matching algorithms, there is far less work on dealing with the matching of hierarchical graphs, i.e., DAGs, in which lower (deeper) levels reflect less saliency. Pelillo et al. [34] provided a solution for the matching of hierarchical trees by constructing an association graph using the concept of connectivity and solving a maximal clique problem in this new structure. The latter problem can be formulated as a quadratic optimization problem and they used the so-called replicator dynamical systems developed in theoretical biology to solve it. Pelillo [33] also generalized the framework for matching free trees and using simple payoff-monotonic dynamics from evolutionary game theory. The problem of matching hierarchical trees has also been studied in the context of edit-distance (see, e.g., [38]). In such a setting, one seeks a minimal set of re-labellings, additions, deletions, merges, and splits of nodes and edges that transform one graph into another.

In recent work [10,11], we presented a framework for many-to-many matching of hierarchical structures, where features and their relations were represented using directed edge-weighted graphs. The method began with transforming the graph into a metric tree. Next, using graph embedding techniques, the tree was embedded into a normed vector space. This two-step transformation allowed us to reduce the problem of many-to-many graph matching to a much simpler problem of matching weighted distributions of points in a normed vector space. To compute the distance between two weighted distributions, we used a distribution-based similarity measure, known as the Earth Mover's Distance under transformation [6,37].

As mentioned in Section 1, object categorization has received renewed attention from the recognition community. In [14], Fergus et al. present a method to learn and recognize object class models from unlabeled cluttered scenes in a scale invariant manner. They deploy a probabilistic representation to simultaneously model shape, appearance, occlusion, and relative scale. They also use expectation maximization for learning the parameters of the scale-invariant object model and use this model in a Bayesian manner to classify images. Fei-Fei et al. [13] improved this by proposing a method for learning object categories from just a few training images. Their proposed method is based on making use of priors to construct a generative probabilistic model, assembled from object categories which were previously learned.

Lazebnik et al. [21] present a framework for the representation of 3-D objects using multiple composite local affine parts. Specifically, these are 2D configurations of regions that are stable across a range of views of an object, and also across multiple instances of the same object category. These composite representations provide improved expressiveness and distinctiveness, along with added flexibility for representing complex 3-D objects. Leibe and Schiele [23] propose a new database for comparing different methods for object categorization. The database contains high-resolution color images of 80 objects from 8 different categories, for a total of 3280 images and was used to analyze the performance of several appearance and contour based methods. The best reported method for categorization for this database is a combination of both contour- and shape-based methods. This new generation of categorization systems are primarily appearance-based, and therefore not well-equipped to handle within-class deformations due to significant appearance change, articulation, significant changes in minor geometric detail.

The closest integrated framework to that proposed here is due to Shokoufandeh et al. [41], who match multiscale blob representations represented as directed acyclic graphs. The multi-scale description in that work, due to Marsic [29], excluded geometric relations among sibling nodes and did not include ridge features. Moreover, the matching framework had to choose between two algorithms, one targeting structural matching, while the other enforcing both structural and geometrical graph alignment. Our proposed multiscale image representation is far richer in terms of its underlying features, and resembles that of Bretzner and Lindeberg [3], who explored qualitative, multi-scale hierarchies for object tracking. Our matching framework, on the other hand, offers several orders of magnitude less complexity, handles noise more effectively, and can handle both structural and geometrical matching within the same framework.

3 Building a Qualitative Shape Feature Hierarchy

3.1 Extracting Qualitative Shape Features

As mentioned in Section 1, our qualitative feature hierarchy represents an image in terms of a set of blobs and ridges, captured at appropriate scales. The representation is an extension of the work presented in [3]. Blob and ridge extraction is performed using automatic scale selection, as described in previous work (see [25] and [24]). We also extract a third descriptor, called the *win-*

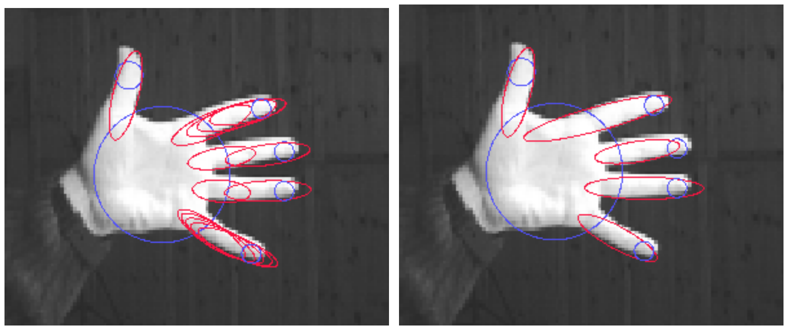

(a) (b)

Fig. 1. Feature Extraction: (a) extracted blobs and ridges at appropriate scales; (b) extracted features before and after removing multiple responses and linking ridges.

dowed second moment matrix, which describes the directional characteristics of the underlying image structure.

A scale-space representation of the image signal f is computed by convolution with Gaussian kernels $g(\cdot;\ t)$ of different variance t, giving $L(\cdot;\ t) = g(\cdot;\ t) * f(\cdot)$. Blob detection aims at locating compact objects or parts in the image. The entity used to detect blobs is the square of the normalized Laplacian operator,

$$\nabla^2_{norm} L = t\,(L_{xx} + L_{yy}). \qquad (1)$$

Blobs are detected as local maxima in scale-space. Figure 1(a) shows an image of a hand with the extracted blobs superimposed. A blob is graphically represented by a circle defining a *support region*, whose radius is proportional to $\sqrt(t)$.

Elongated structures are localized where the multi-scale ridge detector

$$\mathfrak{R}_{norm} L = t^{3/2}\,((L_{xx} - L_{yy})^2 + 4L^2_{xy}) \qquad (2)$$

assumes local maxima in scale-space. Figure 1(a) also shows the extracted ridges, represented as superimposed ellipses, each defining a support region, with width proportional to $\sqrt(t)$. To represent the spatial extent of a detected image structure, a windowed second moment matrix

$$\Sigma = \int\limits_{\eta \in \mathbb{R}^2} \begin{pmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{pmatrix} g(\eta; t_{int})\, d\eta \tag{3}$$

is computed at the detected feature position and at an integration scale t_{int} proportional to the scale t_{det} of the detected image feature. There are two parameters of the directional statistics that we make use of here: the *orientation* and the *anisotropy*, given from the eigenvalues λ_1 and λ_2 ($\lambda_1 > \lambda_2$) and their corresponding eigenvectors $\vec{e}_{\lambda_1}$ and $\vec{e}_{\lambda_2}$ of Σ. The anisotropy is defined as

$$\tilde{Q} = \frac{1 - \lambda_2/\lambda_1}{1 + \lambda_2/\lambda_1}, \tag{4}$$

while the orientation is given by the direction of $\vec{e}_{\lambda_1}$.

To improve feature detection in scenes with poor intensity contrast between the image object and background, we utilize color information. This is done by extracting features in the R, G and B color bands separately, along with the intensity image. Re-occurring features are awarded with respect to significance. Furthermore, if we have advance information on the color of the object, improvements can be achieved by weighting the significance of the features in the color bands differently.

When constructing a feature hierarchy, we extract the N most salient image features, ranked according to the response of the scale-space descriptors used in the feature extraction process. From these features, a *Feature Map* is built according to the following steps:

3.1.1 Merging multiple feature responses.

This step removes multiple overlapping responses originating from the same image structure, the effect of which can be seen in Figure 1(a). To be able to detect overlapping features, a measure of inter-feature similarity is needed. For this purpose, each feature is associated with a 2-D Gaussian kernel $g(\vec{x}, \Sigma)$, where the covariance is given from the second moment matrix. When two features are positioned near each other, their Gaussian functions will intersect. The similarity measure between two such features is based on the *disjunct volume* D of the two Gaussians [20]. This volume is calculated by integrating the square of the difference between the two Gaussian functions (g_A, g_B) corresponding to the two intersecting features A and B:

$$D(A, B) = \sqrt{\frac{|\Sigma_A| + |\Sigma_B|}{2}} \int_{\Re^2} (g_A - g_B)^2 d\mathbf{x}. \tag{5}$$

The disjunct volume depends on the differences in position, variance, scale and orientation of the two Gaussians, and for ridges is more sensitive to translations in the direction perpendicular to the ridge.

3.1.2 Linking ridges

The ridge detection will produce multiple responses on a ridge structure that is long compared to its width. These ridges are linked together to form one long ridge, as illustrated in Figure 1(b). The criteria for when to link two ridges is based on two conditions: 1) they must be aligned, and 2) their support regions must overlap. After the linking is performed, the anisotropy and support region for the resulting ridge is re-calculated. The anisotropy is re-calculated from the new length/width relationship as 1-(*width of structure)/(length of structure).*

Once the Feature Map is constructed, the component features are assembled into a directed acyclic graph. Associated with each node (blob/ridge) are a number of attributes, including position, orientation, and support region. A feature at the coarsest scale of the Feature Map is chosen as the root. Next, finer-scale features that overlap with the root become its children through hierarchical edges. These children, in turn, select overlapping features (again through hierarchical edges) at finer scales to be their children, etc. ¿From the unassigned features, the feature at the coarsest scale is chosen as a new root. Children of this root are selected from unassigned as well as assigned features, and the process is repeated until all features are assigned to a graph. A child node can therefore have multiple parents. To yield a single rooted graph, which is needed in the matching step, a virtual top root node is inserted as the parent of all root nodes in the image.

Associated with each edge are a number of important geometric attributes. For an edge $\mathcal{E}$, directed from a vertex $\mathcal{V}_A$ representing feature $\mathcal{F}_A$, to a vertex $\mathcal{V}_B$ representing feature $\mathcal{F}_B$, we define the following attributes, as shown in Figure 2:

- **Distance.** Two measures of inter-feature distance are associated with the edge: 1) the smallest distance d from the support region of $\mathcal{F}_A$ to the support region of $\mathcal{F}_B$, normalized to the the largest of the radii r_A and r_B; and 2) the distance between their centers normalized to the radius r_A of $\mathcal{F}_A$ in the direction of the distance vector between their centers.
- **Relative orientation.** The relative orientation between $\mathcal{F}_A$ and $\mathcal{F}_B$.

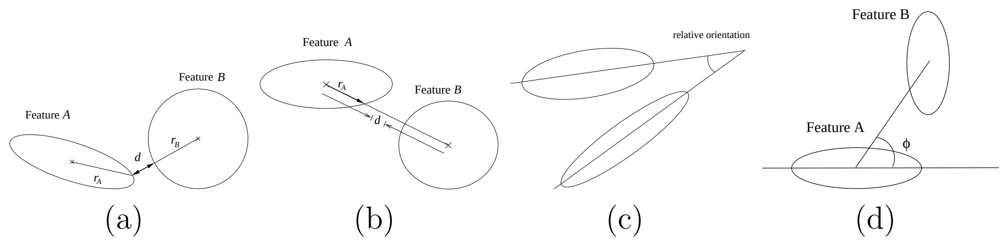


Fig. 2. The four edge relations: (a,b) two normalized distance measures, (c) relative orientation, and (d) bearing

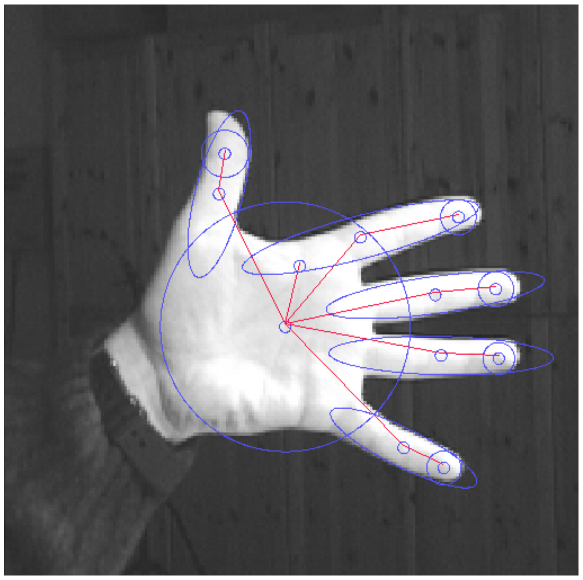


Fig. 3. Example graph of a hand image, with the hierarchical edges shown in green.

- **Bearing.** The bearing of a feature $\mathcal{F}_B$, as seen from a feature $\mathcal{F}_A$, is defined as the angle of the distance vector $x_B - x_A$ with respect to the orientation of A measured counter-clockwise.
- **Scale ratio.** The scale invariant relation between $\mathcal{F}_A$ and $\mathcal{F}_B$ is a ratio between scales $t_{\mathcal{F}_A}$ and $t_{\mathcal{F}_B}$.

An example of a blob graph for a hand image, showing hierarchical edges, is shown in Figure 3.

4 Matching Problem Formulation

Given two images and their corresponding feature map graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_1, E_1)$, with $|V_1| = n_1$ and $|V_2| = n_2$, we seek a method for computing their similarity. In the absence of noise, segmentation errors, occlusion, and clutter, computing the similarity of G_1 and G_2 could be formulated as a label-consistent graph isomorphism problem. However, under normal imaging conditions, there may not exist significant subgraphs common to G_1 and G_2. We therefore seek an approximate solution that captures both the structural and geometrical similarity of G_1 and G_2 as well as corresponding node similarity. Structural similarity is a domain-independent measure that accounts for similarity in the "shapes" of two graphs, in terms of numbers of nodes, branching factor distributions, etc. Geometrical similarity, on the other hand, accounts for consistency in the relative positions, orientations, and scales of nodes in the two graphs. In the following subsections, we describe these two signatures and combine them in an efficient algorithm to match two blob graphs.

4.1 Encoding Graph Structure

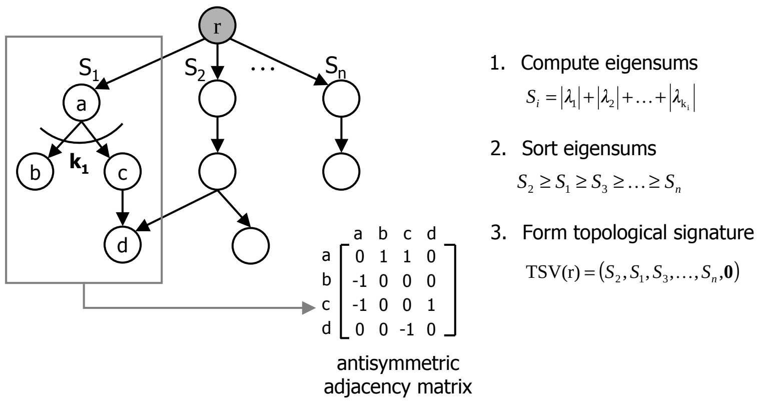
As described in Section 1, our previous work on rooted tree matching drew on the eigenvalues of a tree's symmetric $\{0, 1\}$ adjacency matrix to encode the "shape" of a tree using a low-dimensional vector. The eigenvalues of a graph's adjacency matrix characterize the graph's degree distribution, an important structural property of the graph. In extending that approach to DAG matching, we first draw on our recent work in indexing hierarchical (DAG) structures [40], in which the *magnitudes* of the eigenvalues of a DAG's antisymmetric $\{0, 1, -1\}$ adjacency matrix[1] are used to encode the shape of a DAG using a low-dimensional vector. Moreover, in [40], we show that the eigenvalues are invariant to minor structural perturbation of the graph due to noise and occlusion.

Let's briefly review the construction of our graph abstraction; details can be found in [40]. Let $\mathfrak{D}$ be a DAG whose maximum branching factor is $\Delta(\mathfrak{D})$, and let the subgraphs of its root be $\mathfrak{D}_1, \mathfrak{D}_2, \ldots, \mathfrak{D}_S$, as shown in Figure 4. For each subgraph, $\mathfrak{D}_i$, whose root degree is $\delta(\mathfrak{D}_i)$, we compute the magnitudes of the (complex) eigenvalues of $\mathfrak{D}_i$'s submatrix, sort the magnitudes in decreasing order, and let S_i be the sum of the $\delta(\mathfrak{D}_i)-1$ largest magnitudes. The sorted S_i's become the components of a $\Delta(\mathfrak{D})$-dimensional vector assigned to the DAG's root. If the number of S_i's is less than $\Delta(\mathfrak{D})$, then the vector is padded with zeroes. We can recursively repeat this procedure, assigning a vector to each nonterminal node in the DAG, computed over the subgraph rooted at that node. We call each such vector a *topological signature vector*, or TSV. The TSV assigned to a node allows the structural context of the node (i.e., the subgraph rooted at the node) to be encapsulated in the node as an attribute.

4.2 Encoding Graph Geometry

The above encoding of structure suffers from the drawback that it does not capture the geometry of the nodes. For example, two graphs with identical structure may differ in terms of the relative positions of their nodes, the rela-

[1] A matrix with 1's (-1's) indicating a forward (backward) edge between adjacent nodes in the graph (and 0's on the diagonal).


r
S1
S2
Sn
a
b
c
d
k1
1. Compute eigensums
$S_i = |\lambda_1| + |\lambda_2| + \ldots + |\lambda_{k_i}|$
2. Sort eigensums
$S_2 \geq S_1 \geq S_3 \geq \ldots \geq S_n$
3. Form topological signature
$\mathrm{TSV(r)} = (S_2, S_1, S_3, \ldots, S_n, \mathbf{0})$
antisymmetric adjacency matrix

$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 1 & 0 \\ b & -1 & 0 & 0 & 0 \\ c & -1 & 0 & 0 & 1 \\ d & 0 & 0 & -1 & 0 \end{array}$$

Fig. 4. Forming the structural signature.

tive orientations of their nodes (for elongated nodes), and the relative scales of their nodes. Just as we derived a topological signature vector, which encodes the “neighborhood” structure of a node, we now seek an analogous “geometrical signature vector”, which encodes the neighborhood geometry of a node. This geometrical signature will be combined with our new topological signature in a new algorithm that computes the distance between two directed acyclic graphs and preserves hierarchical and sibling constraints.

Let $G = (V, E)$ be a graph to be recognized (input image). For every pair of vertices $u, v \in V$, if there is an edge $\mathfrak{E} = (u, v)$ between them, we let $R_{u,v}$ denote the attribute vector associated with edge $\mathfrak{E}$. The entries of each such vector represent the set of relations $\mathfrak{R} = \{$distance, relative orientation, bearing, scale ratio$\}$ between u and v, as shown in Figure 5. For a vertex $u \in V$, we let $N(u)$ denote the set of vertices $v \in V$ such that the directed edge (u, v) corresponds to a sibling relation. For a relation $p \in \mathfrak{R}$, we will use $\mathfrak{P}(u, p)$ to denote the distribution of values of relation p between vertex u and all the vertices in the set $N(u)$, i.e., $\mathfrak{P}(u, p)$ is a histogram encoding the p^{th} entry of the vectors $R_{u,v}$ for $v \in N(u)$.[2]

[2] The exception to this rule is the orientation relation. Rather than use absolute

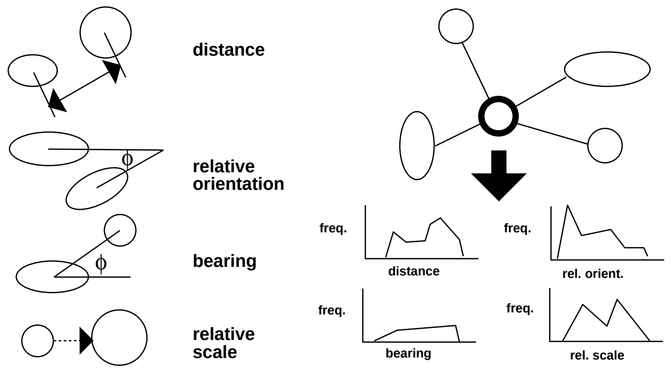


Fig. 5. Forming the geometric signature.

Given two graphs $G = (V, E)$ and $G' = (V', E')$ with vertices $u \in V$ and $u' \in V'$, we compute the similarity between u and u' in terms of their respective distributions $\mathfrak{P}(u, p)$ and $\mathfrak{P}(u', p)$, for all $p \in \mathfrak{R}$. Now, let $d_p(u, u')$ denote the Earth Movers Distance (EMD) [37] between two such distributions $\mathfrak{P}(u, p)$ and $\mathfrak{P}(u', p)$, i.e., $d_p(u, u')$ denotes the minimum amount of work (defined in terms of displacements of the masses associated with histograms $\mathfrak{P}(u, p)$ and $\mathfrak{P}(u', p)$) it takes to transform one distribution into another. The main advantage of using EMD to compute $d_p(u, u')$ lies in the fact that it subsumes many histogram distances and permits partial matches in a natural way. This important property allows the similarity measure to deal with the case where the masses associated with distributions $\mathfrak{P}(u, p)$ and $\mathfrak{P}(u', p)$ are not equal. Details of the method are presented in [19]. Given the the values of $d_p(u, u')$ for all $p \in \mathfrak{R}$, we arrive at a final node similarity function for vertices u and u':

$$\sigma(u, u') = e^{-\sum_{p \in \mathfrak{R}} d_p(u,u')}.$$

orientation, measured with respect to a reference direction, we instead use the angle from the previous edge in a clockwise ordering of edges emanating from a vertex.

As mentioned in Section 1, our previous work addressed the problem of matching rooted trees, and was unable to match directed acyclic graphs, unable to accommodate geometric relations among nodes, and unable to preserve hierarchical and sibling relations. Still, it serves as the starting point for our new algorithm, and we review it accordingly. The method was a modified version of Reyner's algorithm [36,47] for finding the largest common subtree. The main idea of the algorithm was to cast the structural matching problem as a set of bipartite graph matching problems. A similarity matrix between the two graphs' nodes was constructed with each entry computing the pairwise similarity between a particular node in the first tree and a node in the second tree. This similarity measure was a weighted combination of the distance between the two nodes' TSVs, reflecting the extent to which their underlying subtrees had similar structure, and the two nodes' internal attributes. [3]

The algorithm is illustrated in Figure 6. The best pairwise node correspondence obtained after a maximum cardinality maximum weight (MCMW) bipartite matching is extracted and put into the solution set of correspondences. In a greedy fashion, the algorithm recursively matches the two resulting pairs of corresponding forests, at each step computing a maximum matching and placing the best corresponding pair of nodes in the solution set. The key idea in casting a graph-matching problem as a number of MCMW bipartite matching problems is to use the topological signature vectors (TSV) to penalize nodes with different underlying graph structure. This effectively allows us to discard

[3] For shock graphs, each node encoded a set of medial axis, or shock, points and their similarity was computed based on a Hausdorff distance between these point sets.

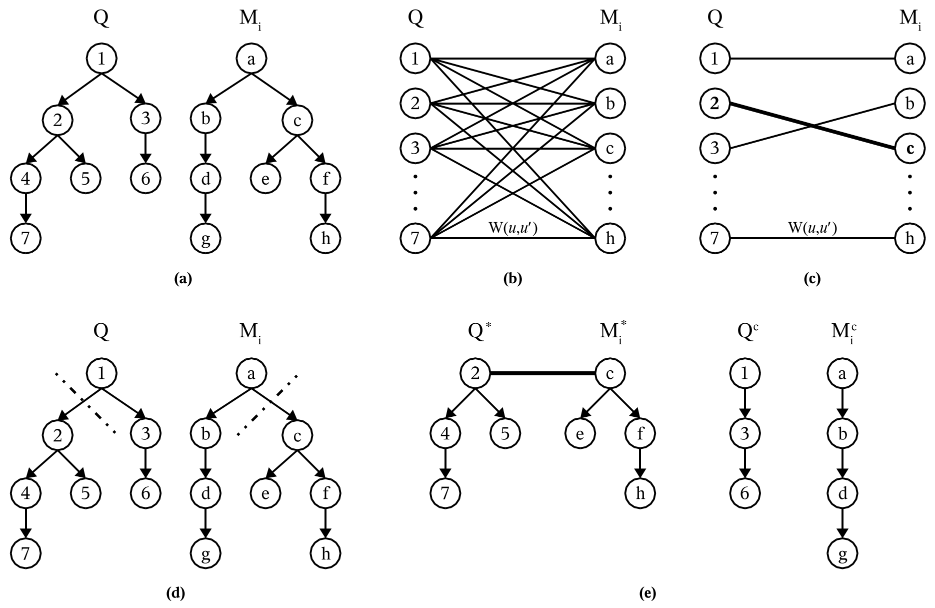


Fig. 6. The DAG matching algorithm. (a) Given a query graph and a model graph, (b) form bipartite graph in which the edge weights are the pair-wise node similarities. Then, (c) compute a maximum matching and add the best edge to the solution set. Finally, (d) split the graphs at the matched nodes and (e) recursively descend.

the graphs' edge structure and formulate the problem as an attributed point matching problem, with a node's underlying structural context encoded as a low-dimensional vector node attribute.

To extend this framework to accommodate DAG matching, geometric relations, and hierarchical/sibling constraint satisfaction, we begin by introducing some definitions and notations. Let $\mathfrak{Q} = (V_{\mathfrak{Q}}, E_{\mathfrak{Q}})$ and $\mathfrak{M} = (V_{\mathfrak{M}}, E_{\mathfrak{M}})$ be the two DAGs to be matched, with $|V_{\mathfrak{Q}}| = n_{\mathfrak{Q}}$ and $|V_{\mathfrak{M}}| = n_{\mathfrak{M}}$. Define d to be the maximum degree of any vertex in $\mathfrak{Q}$ and $\mathfrak{M}$, i.e., $d = \max(\delta(\mathfrak{Q}), \delta(\mathfrak{M}))$. For each vertex v, let $\chi(v) \in R^d$ be the unique topological signature vector (TSV), introduced in Section 4.1. The bipartite edge weighted graph $\mathfrak{G}(V_{\mathfrak{Q}}, V_{\mathfrak{M}}, E_{\mathfrak{G}})$ is represented as a $n_{\mathfrak{Q}} \times n_{\mathfrak{M}}$ matrix $\mathbf{W}$ whose (q, m)-th entry has the value:

$$\mathbf{W}_{q,m} = \alpha \ \sigma(q, m) + (1 - \alpha) \ \left(||\chi(q) - \chi(m)||\right), \tag{6}$$

where $\sigma(q, m)$ denotes the node similarity between nodes $q \in \mathfrak{Q}$ and $m \in \mathfrak{M}$, and α is a convexity parameter that weights the relevance of each term. Using the scaling algorithm of Gabow and Tarjan [15], we can efficiently compute the maximum cardinality, maximum weight matching in $\mathfrak{G}$ with complexity $O(|V||E|)$, resulting in a list of node correspondences between $\mathfrak{Q}$ and $\mathfrak{M}$, called $\mathfrak{L}$, that can be ranked in decreasing order of similarity.

This set of node correspondences maximizes the sum of node similarities, but does not enforce any hierarchical constraints other than the implicit ones encoded in $(||\chi(q) - \chi(m)||)$. Thus, instead of using all the node correspondences, we take a greedy approach and assume that only the first one is correct, and remove the subgraphs rooted at the selected matched pair of nodes. We now have two smaller problems of graph matching, one for the pair of removed subgraphs, and another for the two remainders of the original graphs. Both subproblems can, in turn, be solved by a recursive call of the above algorithm. The complexity of such a recursive algorithm is $O(n^3)$.

It turns out that splitting subgraphs at nodes with high confidence of being a good correspondence is not a strong enough constraint to guarantee that all the hierarchical relations are satisfied. Consider, for example, the graphs in Figure 7. After the first iteration of the matching algorithm, nodes (q_5, m_5) will be matched since their similarity is the highest one in $\mathfrak{L}_1$. In the next iteration, the subgraph rooted at q_5, $\mathfrak{Q}^*$, and the subgraph rooted at m_5, $\mathfrak{M}^*$, as well as their corresponding complement graphs $\mathfrak{Q}^c$ and $\mathfrak{M}^c$, will be recursively evaluated.[4] When matching $\mathfrak{Q}^c$ against $\mathfrak{M}^c$, the best node correspondence according to the outlined algorithm will be (q_4, m_3). It is easy to see that this

[4] In the recursive call for $\mathfrak{Q}^*$ and $\mathfrak{M}^*$, nodes q_5 and m_5 will be in the solution set, and so they will not be evaluated again.

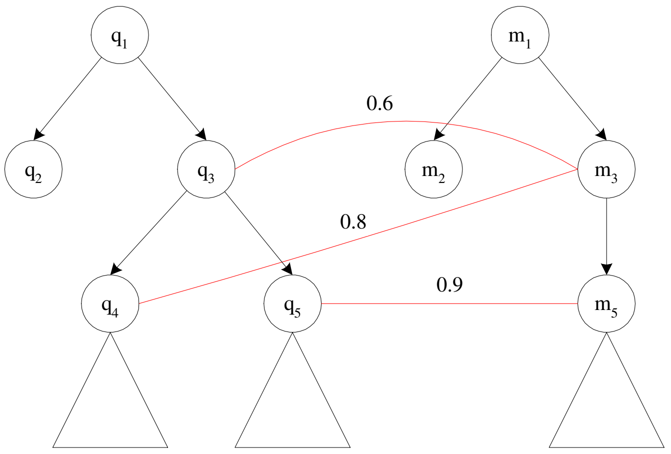


Fig. 7. A case in which the hierarchical constraints between query nodes and model nodes will be violated after two iterations of the algorithm. Note that only non-zero $\mathbf{W}_{q,m}$ values are shown.

match violates the hierarchical constraints among nodes because the siblings q_4 and q_5 are mapped to m_3 and m_5, respectively, with m_3 a parent of m_5.

Another constraint that arises in several domains is that of preserving sibling relationships. Note that this constraint is not, strictly speaking, a hierarchical constraint, since there are no hierarchical dependencies among sibling nodes. While it may be tempting to enforce this constraint when matching, there is a possibility that a sibling relation is genuinely broken by an occluder. In such a case, we may not want to enforce this constraint or else we will be unable to find meaningful matching subgraphs. A compromise solution would be to penalize the matches that break a sibling relationship so as to favor those that provide a good set of correspondences while maintaining these relationships intact.

Figure 8 illustrates the problem. Assuming that (q_5, m_4) has just been added to the solution set, the next best correspondence is (q_4, m_5), which violates a sibling constraint. To avoid this, we can propagate the information provided by the previous best match, (q_5, m_4). This information is used to favor q_4's

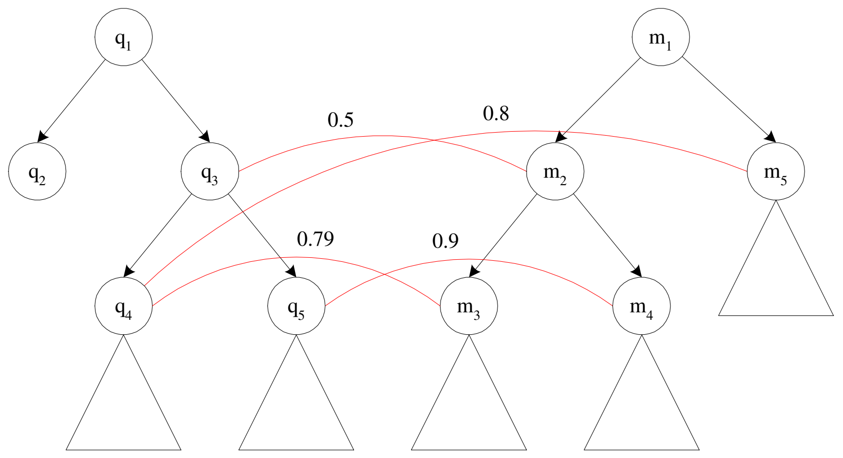


Fig. 8. Preserving sibling relationships by propagating the information from the previous best match. The matching pair (q_4, m_3) is chosen over the slightly better match (q_4, m_5), because it results in two siblings in the query being matched to two siblings in the model.

sibling, so that (q_4, m_3) can be chosen instead. Since we do not want to become too sensitive to noise in the graph, we shall consider preserving the sibling-or-sibling-descendant relationships instead of the stricter sibling relationship. We will refer to this asymmetric relation between nodes as the SSD relation. [5] Note that due to the asymmetry of the relation, the desired propagation of information will occur only whenever the algorithm proceeds in a top-down fashion. In the next section, we will see how to promote a top-down node matching.

Before continuing, let us define the rather intuitive node relationships that we will be working with. Let $\mathfrak{G}(V, E)$ be a DAG and let u, v be two nodes in V. We say that u is a parent of v if there is an edge from u to v. Furthermore, let u be the ancestor of v if and only if there is a path from u to v. Similarly, let u be a SSD of v if and only if there exists a parent of u that is also an ancestor of v.

[5] Note that while the sibling relationship is symmetric, the SSD relationship is not, i.e., if u is the "nephew" of v, then SSD(u,v) is true, but SSD(v,u) is false.

The relations defined above will allow us to express the desired constraints. However, we first need to determine how to make this information explicit, for it is not immediately available from the adjacency matrices of the graphs. A simple method is to compute the transitive closure graphs of our graphs. The transitive closure of a directed graph $\mathfrak{G} = (V, E)$ is a graph $\mathfrak{H} = (V, F)$, with (v, w) in F if and only if there is a path from v to w in $\mathfrak{G}$. The transitive closure can be computed in linear time in the size of the graph $O(|V|\,|E|)$ [16].

¿From the above definition, it is easy to see that the transitive closure of a graph is nothing else than the ancestor relation. Computing the SSD relation, on the contrary, requires a bit of extra work. Let $\mathbf{A}_{\mathfrak{G}}$ be the adjacency matrix of the DAG, $\mathfrak{G}(V, E)$, and let $\mathbf{T}_{\mathfrak{G}}$ be the adjacency matrix of the transitive closure graph of $\mathfrak{G}$. By means of these two matrices, we can now compute the non-symmetric SSD relation by defining $\mathbf{S}_{\mathfrak{G}}$ as the $|V| \times |V|$ matrix, where

$$\mathbf{S}_{\mathfrak{G}}(u, v) = \begin{cases} 1 \text{ if } \exists_{w \in V}\{\mathbf{A}_{\mathfrak{G}}(w, v) = 1 \;\&\; \mathbf{T}_{\mathfrak{G}}(w, u) = 1\}, \\ \\ 0 \text{ otherwise.} \end{cases} \tag{7}$$

Armed with our new matrices $\mathbf{T}_{\mathfrak{Q}}, \mathbf{T}_{\mathfrak{M}}, \mathbf{S}_{\mathfrak{Q}},$ and $\mathbf{S}_{\mathfrak{M}}$, we can update the similarity matrix, $\mathbf{W}$, at each iteration of the algorithm, so as to preserve the ancestor relations and to discourage breaking SSD relations. At the first iteration, $n = 0$, we start with $\mathbf{W}^0 = \mathbf{W}$. Next, let (q', m') be the best node correspondence selected at the n-th iteration of the algorithm, for $n \geq 0$. The new weights for each entry $\mathbf{W}^{n+1}_{q,m}$ of the similarity matrix, which will be used as edge weights in the bipartite graph at iteration $n+1$, are updated according to:

$$\mathbf{W}^{n+1}_{q,m} = \begin{cases} 0 & \text{if } \mathbf{T}_{\mathfrak{Q}}(q,q') \neq \mathbf{T}_{\mathfrak{M}}(m,m'), \\ \beta\ \mathbf{W}^{n}_{q,m} & \text{else if } \mathbf{S}_{\mathfrak{Q}}(q,q') \neq \mathbf{S}_{\mathfrak{M}}(m,m'), \\ \mathbf{W}^{n}_{q,m} & \text{otherwise;} \end{cases} \tag{8}$$

where $0 \leq \beta \leq 1$ is a term that penalizes a pair of sibling nodes in the query being matched to a pair of non-sibling nodes in the model. It is sufficient to apply a small penalty to these cases, since the goal is simply to favor siblings over non-siblings when the similarities of the others are comparable to that of the siblings.

It is clear that when q' and m' are the roots of the subgraphs to match, the ancestor and SSD relations will be true for all the nodes in the DAG. Thus, in practice, when matching the q'-rooted and m'-rooted DAGs, we can avoid evaluating the conditions above. In addition, we know that only a few weights will change as the result of new node correspondence, and so we only need to update those entries of the matrix. This can be done efficiently by designing a data structure that simplifies the access to the weights that are to be updated. Alternatively, the update step can also be efficiently implemented with matrices by noticing that the column of $\mathbf{A}_{\mathfrak{G}}$ corresponding to node u tells us all the parents of u, while the row of $\mathbf{T}_{\mathfrak{G}}$ corresponding to node v give us all the descendants of v. Thus, given a node pair (q', m') and their corresponding $\mathbf{A}_{\mathfrak{Q}}$, $\mathbf{T}_{\mathfrak{Q}}$, $\mathbf{A}_{\mathfrak{M}}$, and $\mathbf{T}_{\mathfrak{M}}$, it is straightforward to select and modify only those entries of $\mathbf{W}^{n}_{q,m}$ that need to be updated at each iteration of the algorithm.

A careful look at the algorithm as it has been stated so far will reveal that, in general, the first node correspondences found will be those among lower-level nodes in the hierarchy. We can expect this bottom-up behavior of the algorithm because the lower-level nodes carry less structural information and so their weight will be less affected by the structural difference of the graphs rooted at them. Therefore, nodes at the bottom of the hierarchy will tend to have high similarity values and consequently, they will be chosen to split the graphs, creating small DAG's with few constraints on the nodes.

A solution to this problem is to redefine the way we choose the best edge from the bipartite matching. Instead of simply choosing the edge with greatest weight, we will also consider the order[6] of the DAG rooted at the matched nodes to select the pair of nodes that have a large similarity weight and are also roots of large subgraphs. We define the mass, $m(v)$, of node v as the order, $n(T)$, of the DAG rooted at v. For a given graph $\mathfrak{G}(V, E)$, the $|V|$-dimensional mass vector, $\mathbf{M}_{\mathfrak{G}}$, in which each of its dimensions is the mass $m(v)$ of a distinct $v \in V$, can be computed from the transitive closure matrix, $\mathbf{T}_{\mathfrak{G}}$, of the graph by $\mathbf{M}_{\mathfrak{G}} = \mathbf{T}_{\mathfrak{G}} \times \vec{1}$, where $\vec{1}$ is the $|V|$-dimensional vector whose elements are all equal to 1. Thus, $\mathbf{M}_{\mathfrak{G}}$ is a vector in which each element $\mathbf{M}_{\mathfrak{G}}(v)$, for $v \in V$, is the number of nodes in the DAG rooted at v.

Unfortunately, the mass does not give us enough information about the depth of the subgraph rooted at a node since, for example, the path of n nodes has the same mass as the star of n nodes. A better idea is to consider the cumulative mass, $\hat{m}$. Let $\hat{m}(v)$ be defined as the sum of all the masses of the nodes of the DAG rooted at v. Thus, the cumulative mass vector will be given

[6] Here we follow the convention in the Graph Theory literature that considers the *order* of a graph to be the number of nodes in the graph, and the *size* of the graph to be the number of edges in the graph.

by $\hat{\mathbf{M}}_{\mathfrak{G}} = \mathbf{T}_{\mathfrak{G}} \times \mathbf{M}_{\mathfrak{G}}$, which can also be written as $\hat{\mathbf{M}}_{\mathfrak{G}} = \mathbf{T}_{\mathfrak{G}}^2 \times \vec{1}$. This vector can then be used to obtain a relative measure of how tall and wide the rooted subgraphs are with respect to the graph they belong to, by simply normalizing the masses. Let $\tilde{\mathbf{M}}_{\mathfrak{G}}$ be the normalized cumulative mass vector given by

$$\tilde{\mathbf{M}}_{\mathfrak{G}} = \frac{\hat{\mathbf{M}}_{\mathfrak{G}}}{\max v \in V \left\{ \hat{\mathbf{M}}_{\mathfrak{G}}(v) \right\}}, \tag{9}$$

where the normalizing factor will correspond to the cumulative mass of the node whose in-degree is zero —a root— and has the greatest cumulative mass in $\mathfrak{G}$.

The cumulative mass is exactly the piece of information we need, since it should be easy to see that for all the trees with n nodes, the star is the one with smallest cumulative mass, while the path is the one with the greatest. Hence, the cumulative mass, $\hat{m}$, for the root of a tree of order n satisfies $2n - 1 \leq \hat{m} \leq \frac{1}{2}n(n+1)$. This measure is a good indicator of how deep and wide a subtree is, and so provides a means to find a compromise between the node similarities and their positions in the graph.

We can then promote a top-down behavior in the algorithm by selecting the match $(q,m)^+$ from the list, $\mathfrak{L}$, returned by each MCMW bipartite matching, with the maximum convex sum of the similarity and the relative mass of the matched nodes,

$$(q,m)^+ = \text{argmax}_{(q,m)\in\mathfrak{L}} \left\{ \gamma \mathbf{W}_{q,m} + (1-\gamma) \max(\tilde{\mathbf{M}}_{\mathfrak{Q}}(q), \tilde{\mathbf{M}}_{\mathfrak{M}}(m)) \right\}, \tag{10}$$

where $0 \leq \gamma \leq 1$ is a real value that controls the influence of the relative cumulative mass in selecting the best match. Since we want to promote a top-down association of nodes without distorting the actual node similarities, we

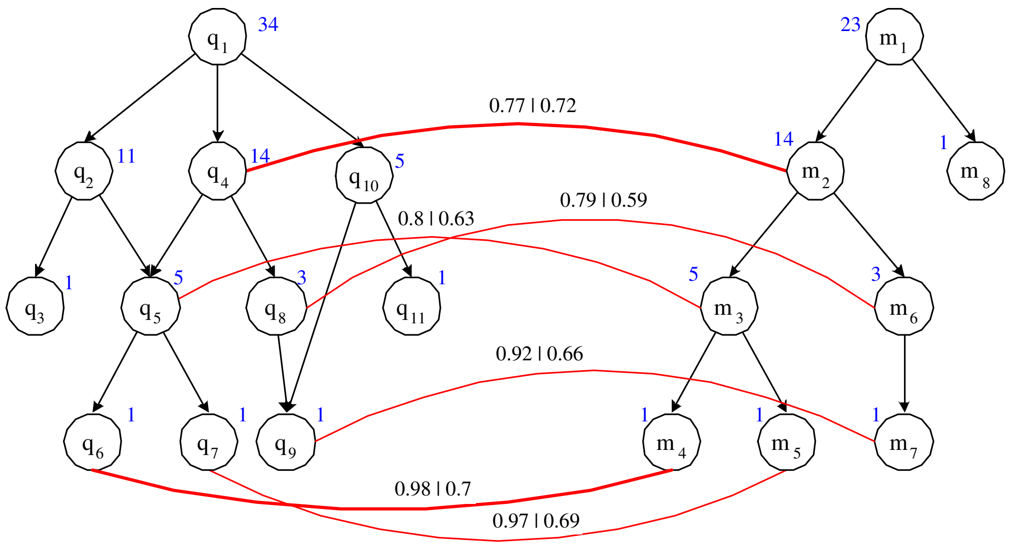


Fig. 9. An example in which $\gamma < 1$ can promote a top-down behavior in the algorithm. The cumulative mass of each node is shown in blue. Edge weights are computed according to Equation 10, for $\gamma = 1$ and $\gamma = 0.7$. For the given set of node similarities and $\gamma = 1$, the best node correspondence at this iteration is the pair of leaves (q_6, m_4), whereas for $\gamma = 0.7$, the best node correspondence is the pair of non-terminal nodes (q_4, m_2).

suggest γ to be in the interval $[0.7, 0.9]$. In Figure 9, we compare the sequences of graph splits using different values for γ. When $\gamma = 1$, we obtain the original equation in [39] that, as can be seen in the figure, tends to produce a bottom-up behavior of the algorithm.

Given the set of node correspondences between two graphs, the final step is to compute an overall measure of graph similarity. The similarity of the query graph to the model graph is given by

$$\sigma_{\Phi}(\mathfrak{Q}, \mathfrak{M}) = \frac{(n_{\mathfrak{Q}} + n_{\mathfrak{M}}) \sum_{(q,m)^+ \in \Phi} \mathbf{W}_{q,m}}{2 n_{\mathfrak{Q}} n_{\mathfrak{M}}}, \tag{11}$$

where $n_{\mathfrak{Q}}$ and $n_{\mathfrak{M}}$ are the orders of the query graph and the model graph, respectively.

The graph similarity is given by a weighted average of the number of matched nodes in the query and in the model, where the weights are given by the node similarity of each matched node pair. If all the query nodes are matched with similarity 1, i.e., their attributes are identical, we have $\sum_{(q,m)^+ \in \Phi} \mathbf{W}_{q,m} = n_{\mathfrak{Q}}$, and so $\sigma_\Phi(\mathfrak{Q}, \mathfrak{M}) = \frac{1}{2}(\frac{n_{\mathfrak{Q}}}{n_{\mathfrak{M}}} + 1)$. Since all query nodes have been matched, we know that $n_{\mathfrak{M}} \geq n_{\mathfrak{Q}}$, and so $\sigma_\Phi(\mathfrak{Q}, \mathfrak{M})$ will be one when all the model nodes are mapped, and less than one otherwise. Therefore, the graph similarity is proportional to the quality of each pair of node correspondences, and inversely proportional to the number of unmatched nodes, both in the query and in the model. Hence, a model that contains the query as a relatively small subgraph is not as good a match as a model for which most of nodes match those of the query graph, and vice versa.

Finally, it should be noted that the relative weighting of the topological and geometric terms in the bipartite graph edge weights need not be constant for all edges. Since each edge spans an image node and a model node, the model can be used to define an a priori weighting scheme that is edge dependent. Thus, if portions of the model were more geometrically constrained (e.g., articulation was prohibited), those model nodes could have a higher weighting on their geometric similarity component. Similarly, portions of the model that were less constrained could have a higher weighting on the topological similarity component. This is a very powerful feature of the algorithm, allowing the incorporation of local model constraints into the matching algorithm.

The final algorithm is shown in Figure 10. The first step of the algorithm is to compute a node similarity matrix, the transitive closure matrices, the sibling matrices, and the node TSVs for both graphs. Assuming a linear algorithm for the pairwise node similarities, the former matrix can be computed in $O(n^3)$. The other matrices can, in turn, be obtained in linear time and in quadratic

```
procedure isomorphism($\mathfrak{Q}$,$\mathfrak{M}$)
    $\Phi(\mathfrak{Q}, \mathfrak{M}) \leftarrow \emptyset$ ;solution set
    compute the $n_{\mathfrak{Q}} \times n_{\mathfrak{M}}$ weight matrix $\mathbf{W}^0$ from Eq. 6
    $\mathbf{T}_{\mathfrak{Q}}$ = compute transitive closure matrix from $\mathbf{A}_{\mathfrak{Q}}$
    $\mathbf{T}_{\mathfrak{M}}$ = compute transitive closure matrix from $\mathbf{A}_{\mathfrak{M}}$
    $\mathbf{S}_{\mathfrak{Q}}$ = compute SSD matrix from $\mathbf{A}_{\mathfrak{Q}}$ and $\mathbf{T}_{\mathfrak{Q}}$
    $\mathbf{S}_{\mathfrak{M}}$ = compute SSD matrix from $\mathbf{A}_{\mathfrak{M}}$ and $\mathbf{T}_{\mathfrak{M}}$
    compute the TSV of each nonterminal node in $\mathfrak{Q}$ and $\mathfrak{M}$
    unmark all nodes in $\mathfrak{Q}$ and in $\mathfrak{M}$
    call match(root($\mathfrak{Q}$),root($\mathfrak{M}$))
    return($\sigma_\Phi(\mathfrak{Q}, \mathfrak{M})$)
end

procedure match($u$,$v$)
    do
        {
        let $\mathfrak{Q}_u \leftarrow u$ rooted unmarked subgraph of $\mathfrak{Q}$
        let $\mathfrak{M}_v \leftarrow v$ rooted unmarked subgraph of $\mathfrak{M}$
        $\mathfrak{L} \leftarrow$ max cardinality, max weight bipartite matching between
                unmarked nodes in $\mathfrak{G}(V_{\mathfrak{Q}_u}, V_{\mathfrak{M}_v})$ with weights from $\mathbf{W}^{n+1}$ (see[15])
        $(u', v') \leftarrow$ choose max weight pair in $\mathfrak{L}$ from Eq. 10
        $\Phi(\mathfrak{Q}, \mathfrak{M}) \leftarrow \Phi(\mathfrak{Q}, \mathfrak{M}) \cup \{(u', v')\}$
        update the similarity matrix $\mathbf{W}^{n+1}$ according to Eq. 8
        mark $u'$
        mark $v'$
        call match($u'$,$v'$)
        call match($u$,$v$)
        }
    while ($\mathfrak{Q}_u \neq \emptyset$ and $\mathfrak{M}_v \neq \emptyset$)
```

Fig. 10. Algorithm for Matching Two Directed Acyclic Graphs

time, respectively. At each iteration of the algorithm, we have to compute a MCMW bipartite matching, sort its output, and update the similarity matrix. The complexity at each step is then determined by that of the bipartite matching algorithm, $O(|V||E|)$, since it is the most complex operation of the three. The number of iterations is bounded by $min(n_{\mathfrak{Q}}, n_{\mathfrak{M}})$, and so the overall complexity of the algorithm is $O(n^3)$. Hence, we have provided the algorithm with important properties for the matching process while maintaining its original complexity. An example of the blob correspondence computed over two hand exemplars is shown in Figure 11.

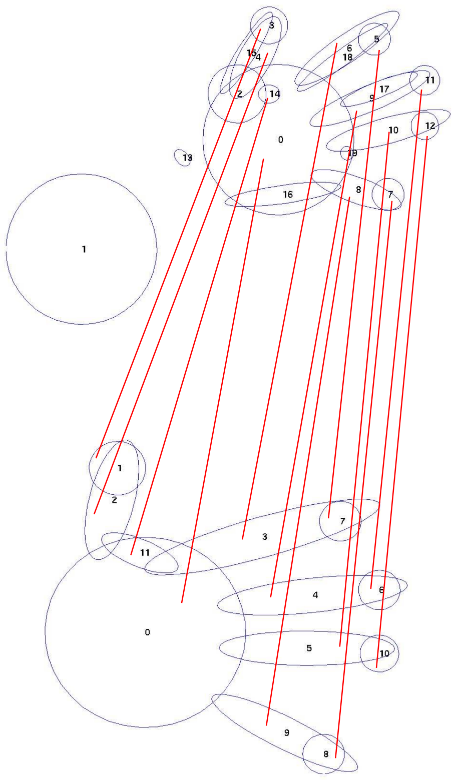


Fig. 11. Example Correspondence Computed between Two Blob Graphs

5 Experiments

We evaluate our framework on the domain of view-based 3-D object recognition where the objective is to choose the right object (identification) for a particular query view and also to determine its correct pose (pose estimation). To provide a comprehensive evaluation, we used two popular image libraries; the Columbia University COIL-20 (20 objects, 72 views per object) [30] and the ETH Zurich ETH-80 (8 categories, 10 exemplars per category, 41 views per exemplar) [22]. Sample views of objects from these two libraries are shown in Figure 12. Note that in the ETH-80 library, some categories have very similar shape, differing only in their appearance. Thus, the horse, dog, and cow categories were collapsed to form a 4-legged animal category, the apple and

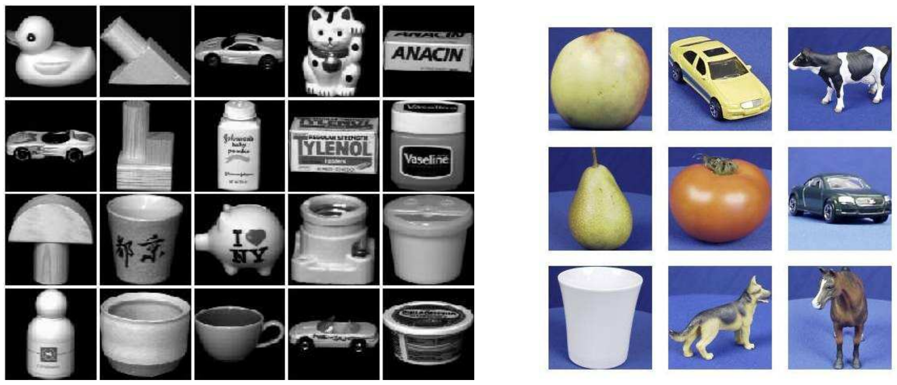


Fig. 12. Views of sample objects from the Columbia University Image Library (COIL-20) and the ETH Zurich (ETH-80) Image Set.

tomato categories were collapsed to form a spherical fruit category, and the two car instance categories were collapsed to form a single car category.

We applied the following procedure to each database to evaluate the proposed framework. We initially removed the first entry from the database, used it as a query, and computed its similarity with each of the remaining views in the database. We then returned the query back to the database and repeated the same process for each of the the other database entries. This process results in a $n \times n$ similarity matrix, where the entry (i, j) indicates how similar views i and j are. For a particular query, we classify its identification as correct if the maximum similarity is obtained with a view which belongs to the same object as the query. Consequently, pose estimation is correct if view i of object j, $v_{i,j}$ matches most closely with $v_{n,j}$, where n is one of i's neighboring views.

Our overall recognition rates for COIL-20 and ETH-80 datasets are 93.5% and 97.1%, respectively. We show a part of the matching results in Table 1. Upon investigation as to why the COIL-20 dataset yields poorer performance, we found that most of the mismatches were between three different car objects: column three of the first row, column one of the second row, and column four of the fourth row, as shown in the left of Figure 12. Despite being different

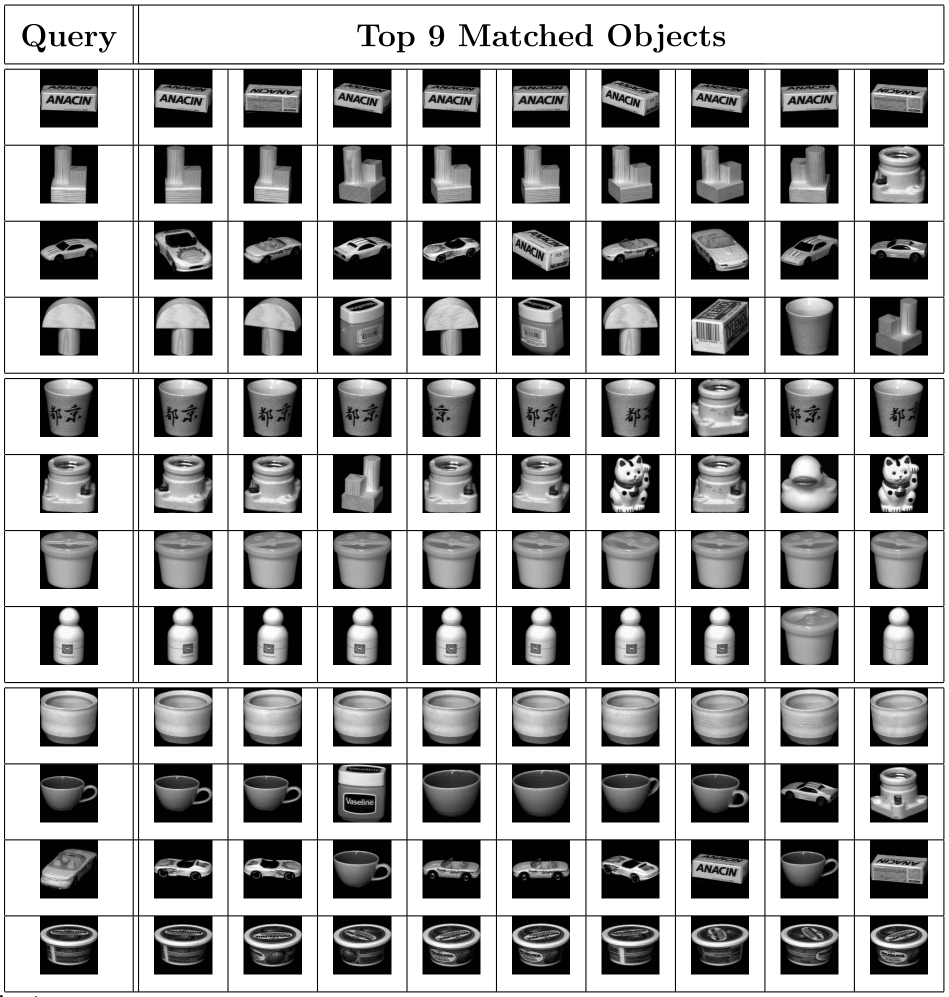


Table 1
Top matched models are sorted by the similarity to the query.

objects with different appearance, their coarse shape structure is similar and their blob graphs are indeed similar. If we group these three exemplars into the same category and count these matches as correct, our recognition rate rises to 96.5%. Our recognition framework is clearly suited to coarse shape categorization as opposed to exemplar matching.

For pose estimation, we observe that in all but 9.8% and 14.6% of the COIL-20 and ETH-80 experiments, respectively, the closest match selected by our algorithm was a neighboring view. Note that if the closest matching view was not an immediate neighbor drawn from the *same exemplar*, the match was deemed incorrect, despite the fact that the matching view might be a neighboring view of a different exmeplar from the same category. This is perhaps overly harsh, as reflected by the 14.6% results, but view alignment between exemplars belonging to the same category was not provided. These results can be considered worst case for several additional reasons. Given the high similarity among neighboring views, it could be argued that our pose estimation criterion is overly harsh, and that perhaps a measure of “viewpoint distance”, i.e., “how many views away was the closest match” would be less severe. In any case, we anticipate that with fewer samples per object, neighboring views would be more dissimilar, and our matching results would improve. More importantly, many of the objects are rotationally symmetric, and if a query has an identical view elsewhere in the dataset, that view might be chosen (with equal similarity) and scored as an error.

To demonstrate the framework's robustness, we performed five perturbation experiments on both datasets. The experiments were identical to the experiments above, except that we randomly chose a node, v, in the query graph, if the number of nodes in the directed acyclic subgraph rooted at v was less than 10% of the number of nodes in the original graph, we deleted the rooted subgraph from the query. We then repeated the same process for maximum ratios of 20%, 30%, 40%, and 50%. The results are shown in Table 2, and reveal that the error rate increases gracefully as a function of increased perturbation. Although not a true occlusion experiment, which would require that we replace the removed subgraph with an occluder subgraph, these results demonstrate the framework's ability to match local structure, a property essential for handling occlusion.

Perturbation	10%	20%	30%	40%	50%
Recognition rate (COIL-20)	91.2%	89.5%	87.3%	83.7%	78.6%
Recognition rate (ETH-80)	94.2%	91.5%	89.3%	84.7%	82.6%

Table 2
Recognition rate as a function of increasing perturbation in the form of missing data. Percentages indicate how much of the query graph was removed prior to matching.

6 Limitations

Both the representation and matching components of our integrated framework have limitations. Since it is based on image gradients, the blob and ridge decomposition does not perform well in the presence of textured surfaces, and spurious and missing blobs may result. Although the matching algorithm can accommodate both noise and occlusion, it does rely on there being a sufficient number of one-to-one correspondences to discriminate the correct model from other models. If blobs are highly over- or under-segmented, matching may fail as too few one-to-one correspondences may exist.

7 Conclusions

Matching two images whose similarity exists at the coarse shape level is critical to object categorization. Blobs and ridges provide an ideal multiscale part vocabulary for coarse shape modeling which, when combined with an array of geometric relations in the form of a graph, yield a powerful categorical shape representation. provide a powerful, hierarchical characterization of an object's coarse shape. Our inexact graph matching framework exploits both the topological as well as the geometrical relations in a directed acyclic graph to yield an efficient algorithm for coarse-to-fine shape matching. We have demonstrated the generality of the framework by applying it to three different domains (without domain-specific tuning), with very encouraging results in each domain.

8 Acknowledgements

The authors gratefully acknowledge the support of the ONR, NSERC, NSF, CITO, NSERC-IRIS, CITO, and MD Robotics.

References

[1] G. Agin and T. Binford. Computer description of curved objects. *IEEE Transactions on Computers*, C-25(4):439–449, 1976.

[2] T. Binford. Visual perception by computer. In *Proceedings, IEEE Conference on Systems and Control*, Miami, FL, 1971.

[3] L. Bretzner and T. Lindeberg. Qualitative multi-scale feature hierarchies for object tracking. *Journal of Visual Communication and Image Representation*, 11:115–129, 2000.

[4] R. Brooks. Model-based 3-D interpretations of 2-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):140–150, 1983.

[5] Peter J. Burt. Attention Mechanisms for Vision in a Dynamic World. In *Proceedings of the International Conference on Pattern Recognition, Vol.1*, pages 977–987, The Hague, The Netherlands, 1988.

[6] S. D. Cohen and L. J. Guibas. The earth mover's distance under transformation sets. In *Proceedings, 7th International Conference on Computer Vision*, pages 1076–1083, Kerkyra, Greece, 1999.

[7] J. Crowley and A. Parker. A representation for shape based on peaks and ridges in the difference of low-pass transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(2):156–169, March 1984.

[8] James L. Crowley. A Multiresolution Representation for Shape. In Azriel Rosenfeld, editor, *Multiresolution Image Processing and Analysis*, pages 169–189. Springer Verlag, Berlin, 1984.

[9] James L. Crowley and Arthur C. Sanderson. Multiple Resolution Representation and Probabilistic Matching of 2–D Gray–Scale Shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):113–121, January 1987.

[10] M. Fatih Demirci, A. Shokoufandeh, S. Dickinson, Y. Keselman, and L. Bretzner. Many-to-many matching of scale-space feature hierarchies using metric embedding. In *Proceedings, Scale Space Methods in Computer Vision, 4th International Conference*, pages 17–32, June 2003.

[11] M. Fatih Demirci, A. Shokoufandeh, S. Dickinson, Y. Keselman, and L. Bretzner. Many-to-many feature matching using spherical coding of directed graphs. In *Proceedings, 8th European Conference on Computer Vision*, pages 332–335, May 2004.

[12] S. Dickinson, M. Pelillo, and R. Zabih. Special section on graph algorithms and computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10), October 2001.

[13] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples. In *Workshop on Generative-Model Based Vision*, 2004.

[14] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference*

on Computer Vision and Pattern Recognition, volume 2, pages 264–271, june 2003.

[15] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for general graph matching problems. *Journal of the ACM*, 38:815–853, 1991.

[16] A. Goralcikova and V. Konbek. A reduct and closure algorithm for graphs. *Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science 74:301–307, 1979.

[17] Martin Jägersand. Saliency Maps and Attention Selection in Scale and Spatial Coordinates: An Information Theoretic Approach. In *Proceedings of the 5th International Conference on Computer Vision*, pages 195–202, Boston, MA, June 1995.

[18] Y. Keselman and S. Dickinson. Generic model abstraction from examples. *IEEE PAMI*, 27(7), 2005.

[19] Y. Keselman, A. Shokoufandeh, M. Demirci, , and S. Dickinson. Many-to-many graph matching via metric embedding. In *Proceedings, IEEE CVPR*, Madison, WI, 2003.

[20] I. Laptev and T. Lindeberg. Tracking of multi-state hand models using particle filtering and a hierarchy of multi-scale image features. In *Proc. Scale-Space'01*, Vancouver, Canada, Jul. 2001.

[21] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Learning local affine-invariant part models for object class recognition. In *Workshop on Learning, Snowbird, Utah*, 2004.

[22] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Madison, WI, June 2003.

[23] Bastian Leibe and Bernt Schiele. Analyzing appearance and contour based methods for object categorization. In *CVPR (2)*, pages 409–415, 2003.

[24] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *Int. J. of Computer Vision*, 30:117–154, 1998.

[25] T. Lindeberg. Feature detection with automatic scale selection. *Int. J. of Computer Vision*, 30:77–116, 1998.

[26] Tony Lindeberg. Detecting Salient Blob–Like Image Structures and Their Scales With a Scale–Space Primal Sketch—A Method for Focus–of–Attention. *International Journal of Computer Vision*, 11(3):283–318, December 1993.

[27] Stephane Mallat and Wen Liang Hwang. Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, 38(2):617–643, March 1992.

[28] D. Marr and H. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Royal Society of London*, B 200:269–294, 1978.

[29] Ivan Marsic. Data–Driven Shifts of Attention in Wavelet Scale Space. Technical Report CAIP–TR–166, CAIP Center, Rutgers University, Piscataway, NJ, September 1993.

[30] H. Murase and S. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[31] R. Nevatia and T. Binford. Description and recognition of curved objects. *Artificial Intelligence*, 8:77–98, 1977.

[32] Bruno Olshausen, Charles Anderson, and David Van Essen. A Neurobiological Model of Visual Attention and Invariant Pattern Recognition Based on Dynamic Routing of Information. *Journal of Neurosciences*, 13(11):4700–4719, November 1992.

[33] M. Pelillo. Matching free trees, maximal cliques, and dynamic monotone game dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1535–1541, November 2002.

[34] M. Pelillo, K. Siddiqi, and S. Zucker. Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1105–1120, November 1999.

[35] Rajesh P.Ñ. Rao, Gregory J. Zelinsky, Mary M. Hayhoe, and Dana H. Ballard. Modeling Saccadic Targeting in Visual Search. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 830–836. MIT Press, Cambridge, MA, 1996.

[36] S. W. Reyner. An analysis of a good algorithm for the subtree problem. *SIAM J. Comput.*, 6:730–732, 1977.

[37] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. of Computer Vision*, 40(2):99–121, 2000.

[38] Thomas B. Sebastian, Philip N. Klein, and Benjamin B. Kimia. Recognition of shapes by editing shock graphs. In *IEEE International Conference on Computer Vision*, pages 755–762, 2001.

[39] A. Shokoufandeh and S. Dickinson. A unified framework for indexing and matching hierarchical shape structures. In *Proceedings, 4th International Workshop on Visual Form*, Capri, Italy, May 28–30 2001.

[40] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, , and S. Zucker. Indexing hierarchical structures using graph spectra. *IEEE PAMI*, 27(7), 2005.

[41] A. Shokoufandeh, I. Marsic, and S. Dickinson. View-based object recognition using saliency maps. *Image and Vision Computing*, 17(5-6):445–460, 1999.

[42] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 30:1–24, 1999.

[43] Eero P. Simoncelli, William T. Freeman, Edward H. Adelson, and David J. Heeger. Shiftable Multiscale Transforms. *IEEE Transactions on Information Theory*, 38(2):587–607, March 1992.

[44] Barnabas Takaĉs and Harry Wechsler. A Dynamic and Multiresolution Model of Visual Attention and Its Application to Facial Landmark Detection. *Computer Vision and Image Understanding*, (in press).

[45] J. Triesch and C. von der Malsburg. Robust classification of hand postures against complex background. In *Proc. Int. Conf. on Face and Gesture Recognition*, pages 170–175, Killington, Vermont, Oct. 1996.

[46] J. Tsotsos. An inhibitory beam for attentional selection. In L. Harris and M. Jenkin, editors, *Spatial Vision in Humans and Robots*. Cambridge University Press, 1993.

[47] Rakesh Verma and Steven W. Reyner. An analysis of a good algorithm for the subtree problem, corrected. *SIAM Journal on Computing*, 18(5):906–908, October 1989.

[48] Laurenz Wiskott, Jean-Marc Fellous, Norbert Krüger, and Christoph von der Malsburg. Face Recognition by Elastic Bunch Graph Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, July 1997.